Int. J. on Recent Trends in Engineering and Technology, Vol. 9, No. 1, July 2013

Implementation of Energy Efficient Scalar Point Multiplication Techniques for ECC

Ravi Kishore Kodali[1], Kashyapkumar H. Patel and Prof. Narasimha Sarma, N.V.S.
[1] National Institute of Technology/ECE, Warangal, India
Email: ravikkodali@gmail.com

***Abstract*—Elliptic curve cryptography (ECC) is mainly an alternative to traditional public-key cryptosystems (PKCs), such as RSA, due to its smaller key size with same security level for resource-constrained networks. The computational efficiency of ECC depends on the scalar point multiplication, which consists of modular point addition and point doubling operations. The paper emphasizes on point multiplication techniques such as Binary, NAF, w-NAF and different coordinate systems like Affine and Projective (Standard Projective, Jacobian and Mixed) for point addition and doubling operations. These operations are compared based on execution time. The results given here are for general purpose processor with 1:73 GHz frequency. The implementation is done over NIST-recommended prime fields 192/224/256/384/521.**

***Index Terms*—NIST, Binary method, NAF, w-NAF, Projective coordinate system, Jacobian coordinate system and Mixed coordinate system.**

I. Introduction

The present and future of both wired and wireless communication applications are more conscious about security issues. It encourages implementation and usage of public key cryptosystems. Specially, Elliptic Curve Cryptography (ECC) [1] attracts attention due to its security level for smaller key length compared to existing public key algorithms such as Rivest-Shamir-Adleman (RSA). The underlying strength of ECC is the hard elliptic curve discrete logarithm problem (ECDLP), which takes exponential time, whereas the RSA takes sub-exponential time. For guaranteed security, RSA require 1024 bits, alternatively ECC takes only 160 bits [2] [3]. The computational efficiency of ECC depends on the chosen appropriate elliptic curve and how fast the scalar point multiplication operations are computed over a prime field.

The core part of ECC is the scalar point multiplication. For example, any scalar, k, prime field, F_p and elliptic curve point, $P \ 2 \ E_p$, the scalar point multiplication is given by $k \ ¤ \ P$. There are different algorithms to represent the scalar value and different coordinate systems to perform the point addition and point doubling operations. These operations are compared computationally due to a trade-off between security level and computational performance. All ECC operations are computed within the prime field $GF(F)$ as well as binary field $GF(F_{2^m})$. The software implementation of point multiplication of 163- bit in 13:9 second on 8- bit processor with 8 MHz clock is given in [4]. An improvement of the work is given [5], which achieved 163- bits multiplication in 0:75 s. The hardware/software co-design using FPGA hardware given in [6]. Moreover the ECC implementation on 8-bit AVR based RISC processor is discussed in [7] for the 160- bit.

Table I. NIST-Recommended Prime Field, $GF(p)$[8].

Prime ¯eld F_p	Numerical Value
F_{192}	$2^{192} \ ¡ \ 2^{64} \ ¡ \ 1$
F_{224}	$2^{224} \ ¡ \ 2^{96} + 1$
F_{256}	$2^{256} \ ¡ \ 2^{224} + 2^{192} + 2^{96} \ ¡ \ 1$
F_{384}	$2^{384} \ ¡ \ 2^{128} \ ¡ \ 2^{96} + 2^{32} \ ¡ \ 1$
F_{521}	$2^{521} \ ¡ \ 1$

Due to easier modular reduction, here we are mainly focusing on prime field operations for standard NIST-recommended prime fields [8] as given in table 1. A prime field is not the only solution for ECC implementation, but NIST has also recommended the binary field $GF(F_{2^m})$, which is relatively more efficient for hardware implementation.

II. Ecc Background

For the prime field $p > 3$ the basic elliptic curve is given by equation 1 [9].

$$E_p(a;b) : y^2 = x^3 + ax + b \ (\mathrm{mod}\ p); \qquad (1)$$

where a and b are constants satisfying $4a^3 + 27b^2 \ 6 \ 0$. All the operations are computed with in the field, 0 to $(p \ ¡ \ 1)$. Higher prime fields give more security level but unsuitable for memory constrained environments. There is a trade-off between security level and memory as well as timing requirements.

For any scalar, $k \ 2 \ F_p$ and any point, $P \ 2 \ E_p(a;b)$, the scalar point multiplication, $k \ ¤ \ P$ is addition of the point, P with itself $(k \ ¡ \ 1)$ times $\underbrace{P + P + ::: + P}_{k \ ¡ \ 1 \ \mathrm{times}}$. At each stage, the point multiplication requires point addition and point doubling operations. Let for any two points, $P(x_1; y_1)$ and $Q(x_2; y_2)$ on the elliptic curve, there is a point $R(x_3; y_3) \ 2 \ E_p(a;b)$, such that $P + Q = R$ and point doubling $2P = R$ is computed in the Affine coordinate system and the same is given in equation 2 [9] [10].

DOI: 01.IJRTET.9.1.525

Point addition	Point doubling
$x_3 = ¢^2 - x_1 - x_2$	$x_3 = ¢^2 - 2x_1$
$y_3 = ¢(x_1 - x_3) - y_1$	$y_3 = ¢(x_1 - x_3) - y_1$
$¢ = \frac{y_2 - y_1}{x_2 - x_1}$	$¢ = \frac{3x_1^2 + a}{2y_1}$

(2)

Here ¢ (slope), requires the modular inversion operation, which is highly computationally intensive and magnified by projective coordinate system is discussed in section IV.

III. Proposed Techniques For Point Multiplication

As mentioned, the scalar point multiplication is core part for ECC implementation. Here, we discuss different algorithms to represent the scalar and to compute the scalar point multiplication.

A. Binary method

In this method, any scalar, $k \in F_p$, is represented in its binary equivalent by $k = \sum_{i=0}^{m-1} k_i * 2^i$, where $\{k_{m-1}, k_{m-2}, ..., k_1, k_0\}$ is its binary form and m is the length of the scalar [11]. For a point, $P \in E_p(a, b)$ and scalar k, such that $Q = k * P$. If 'A' and 'D' are addition and doubling operations, respectively, the total average computational cost = $(\frac{m}{2})A + (m)D$

B. Non-Adjacent Form (NAF) method

As seen in binary method number of point doubling operations is not reducible but the number of point addition operations depends on non-zero elements, so, by reducing non-zero elements indirectly reducing the number of addition operations. If $P = (x, y) \in E_p(a, b)$, then negative of P is given by $-P = (x, p - y)$. Thus subtraction of two points on an elliptic curve is just as efficient as addition. The NAF method uses a signed digit representation for scalar. Let for any scalar, $k \in F_p$ is represented in its NAF form by $k = \sum_{i=0}^{m-1} k_i * 2^i, k_i \in \{-1, 0, 1\}$ [12]. There is unique canonical representation for each scalar k, which consists of digits $\{-1, 0, 1\}$. The point multiplication using NAF method, for each digit requires point doubling operation and if $k_i = -1$ and $k_i = 1$, then additionally point subtraction and point addition operation is performed respectively.

$$\begin{cases} -1 : R = 2R - P \\ 0 : R = 2R \\ 1 : R = 2R + P \end{cases}$$

As per characteristics of the NAF, representation of scalar k, with length m, requires at most one digit extra compared to its binary form with less number of non-zero digits, average density of non-zero digit is $\frac{m}{3}$. It is denoted by NAF(k). Using NAF method for point multiplication, the average computation cost incurred is = $(\frac{m}{3})A + (m)D$.

C. Windowed-NAF method

Between the binary method and the NAF method, the NAF is superior to binary, due to the reduced number of non-zero digits in its NAF representation, which indirectly results in reduced number of addition operations. The w-NAF is an extension of NAF method for different window sizes. Any scalar, $k \in F_p$, is represented in its w-NAF and is given by $k = \sum_{i=0}^{m-1} k_i 2^i, |k_i| < 2^{w-1}$, as shown in algorithm 1 [11].

Algorithm 1 Computing the width-w NAF of a positive Integer

INPUT: Window width w, $k \in Fp$
OUTPUT: $NAF_w(k)$
1) $i \leftarrow 0, r \leftarrow 0$
2) **While** $k > 1$ **do**
  a) **If** k is odd **then**
    i) $r = k \bmod 2^w$,
    ii) **If** $r < 2^{w-1}$ **then** $k_i = r$
    iii) **Else** $k_i = r - 2^w$
  b) **Else** $k_i = 0$
  d) $k = k + k_i$
  e) $k = k/2$; $i = i + 1$
3) **End while**
4) **Return** $(k_{i-1}, k_{i-2}, ..., k_1, k_0)$

As per the characteristics of w-NAF, there is a unique w-NAF representation for scalar k, with length m, the length of scalar representation is at most one digit extra compared to its binary form, but the total average density of non-zero digits is $\frac{m}{w+1}$, which depends on the window-size (w). This method requires the number of pre-computations, given by $(2^{w-2} - 1)$ and these pre-computations are: $\{3, 5, ..., 2^{w-1} - 1\}$. For the example, $w = 5$ the total pre-computation is $2^{5-2} - 1 = 7$ and those are: $\{3P, 5P, 7P, 9P, 11P, 13P, 15P\}$.

Algorithm 2 Window NAF method for point multiplication

INPUT: $k = \{ki-1, ki-2, ..., k1, k0\}$ and $P \in Ep(a, b)$
OUTPUT: $Q = k * P$
1) Compute $Pi = i * P$ for $i = \{1, 2, 3, \cdots, 2w-1 - 1\}$.
2) $Q \leftarrow \infty, i \leftarrow 1 - 1$
3) **For** $i = 0$
  a) $Q \leftarrow 2Q$
  b) **if** $ki = 0$ **then**
    i) **if** $ki > 0$ **then** $Q \leftarrow Q + Pi$
    ii) **else** $Q \leftarrow Q - Pi$
4) **end for**
5) **Return** (Q)

DOI: 01.IJRTET.9.1.525

TABLE II. EXAMPLE OF SCALAR REPRESENTATION

Method	Representation for k = 1234554321	D	A	Pre-computations
Binary	1 0 0 0 1 0 1 1 1 0 1 1 0 0 1 1 1 0 1 0 1 0 0 1 1 0 0 1 0 0 1	31	16	0
NAF	1 0 0 1 0 1 0 -1 0 1 0 -1 0 -1 0 0 -1 0 1 0 0 -1 0 0 -1 0 1 0 0 0 1	31	13	0
$NAF_3(k)$	1 0 0 1 0 0 0 3 0 0 1 0 0 3 0 0 0 0 -3 0 0 -1 0 0 0 0 -3 0 0 0 1	31	9	1
$NAF_4(k)$	1 0 0 0 -7 0 0 0 3 0 0 0 0 5 0 0 0 7 0 0 0 0 7 0 0 0 0 -3 0 0 0 1	32	8	3
$NAF_5(k)$	5 0 0 0 0 -13 0 0 0 0 0 11 0 0 0 0 0 0 -13 0 0 0 0 15 0 0 0 0 -15	29	6	7

As window size is increased, there is a drastic reduction of non-zero digits, which reduces the number of point addition to carry out point multiplication operation. The point multiplication using w-NAF is carried out as per the algorithm 2 [12]. The total average computation is given in equation 3.

$$\text{Total cost} = \underbrace{[1D + (2^{w-2} - 1)A]}_{\text{pre-computations}} + \frac{m}{w+1}A + mD \quad (3)$$

As discussed, among various techniques for point multiplication, w-NAF is superior compared to others, however, it requires pre-computations overhead. Hence, an appropriate window size is to be chosen as per memory requirements. All these methods are compared with an example for a scalar, k and the comparison is given in table II.

IV. COORDINATE SYSTEMS

As discussed in the previous section, point doubling and addition given by equations 2 require compute intensive inversion operations. To eliminate the modular inversion operation, various coordinate systems are considered and are evaluated to carry out these operations.

A. Overview of projective coordinate system

For a field, p and positive integers, s and t, projective coordinates can be defined as an equivalence relation » on the set p^3-(0; 0; 0) of non-zero triples over p field by $(X_1; Y_1; Z_1) \text{ » } (X_2; Y_2; Z_2)$, if $X_1 = \lambda^s X_2; Y_1 = \lambda^t Y_2; Z_1 = \lambda Z_2$ for any, $\lambda \in p^*$, where p^* represent set of non-zero element of p [13]. The above relation can also be represented as

$$(X : Y : Z) = (\lambda^s X; \lambda^t Y; \lambda Z); \lambda \in P^*$$

$(X : Y : Z)$ is called a projective point, and $(X; Y; Z)$ is called the representative of $(X : Y : Z)$. The projective form of the Weierstras elliptic curve equation defined over p is obtained by replacing $x = \frac{X}{Z^s}$ and $y = \frac{Y}{Z^t}$.

B. Standard projective system and mixed coordinate system

For $s = 1$ and $t = 1$, the projective coordinate system is called the standard projective system and the corresponding elliptic curve equation is given equation 4.

$$Y^2Z = X^3 + aXZ^2 + b. \quad (4)$$

For a standard projective coordinate point, $Z \neq 0$, the corresponding affine point is given by $(\frac{X}{Z}, \frac{Y}{Z})$. The point at infinity ∞ is given by $(0 : 1 : 0)$ and negative point of $(X : Y : Z)$ is given by $(X : -Y : Z)$.

C. Point addition using standard projective

For $P(X_1 : Y_1 : Z_1)$ and $Q(X_2 : Y_2 : Z_2)$ in the standard projective coordinate system, $R(X_3 : Y_3 : Z_3) = P + Q$ over $E_p(a, b)$ is given by [14]

$$\left.\begin{aligned} X_3 &= BC \\ Y_3 &= A(B^2X_1Z_2 - A) - B^3Y_1Z_2 \\ Z_3 &= B^3Z_1Z_2 \end{aligned}\right\}, \quad (5)$$

where $A = Y_2Z_1 - Y_1Z_2, B = X_2Z_1 - X_1Z_2,$
$C = A^2Z_1Z_2 - B^3 - 2B^2Y_1Z_2$
Total Addition cost $= \mathbf{12M + 2S}$, where M and S represent multiplication and squaring operations, respectively.

D. Point doubling using standard projective system:

Given a point, $P = (X_1 : Y_1 : Z_1)$ in standard projective coordinates over $E(F_p) : y^2 = x^3 + ax + b$, $2P = (X_3 : Y_3 : Z_3)$ is given by equation 6 [14]

$$\left.\begin{aligned} X_3 &= 2BD \\ Y_3 &= A(4C - D) - 8B^2Y_1^2 \\ Z_3 &= 8B^3 \end{aligned}\right\}, \quad (6)$$

where $A = aZ_1^2 + 3X_1^2, B = Y_1Z_1, C = X_1Y_1B,$
$D = A^2 - 8C$
Total doubling cost .

E. Jacobian coordinate system

For and , the projective coordinate system is called the Jacobian coordinate system and the corresponding elliptic curve equation is given in equation 7.

$$Y^2 = X^3 + aXZ^4 + bZ^6 \quad (7)$$

The Jacobian point (X, Y, Z), $Z \neq 0$ corresponds to the affine point $(\frac{X}{Z^2}, \frac{Y}{Z^3})$. The point at infinity ∞ is given by $(1 : 1 : 0)$ and negative point of $(X : Y : Z)$ is given by

DOI: 01.IJRTET.9.1.525

$(X : -Y : Z)$. In this system, the point doubling formulae for $Q(X_1, Y_1, Z_1)$ are given by equation 8.[11]

$$\left.\begin{array}{l} X_3 = (3X_1^2 + aZ_1^4)^2 - 8X_1Y_1^2 \\ Y_3 = (aX_1^2 + aZ_1^4)(4X_1Y_1^2 - X_3) - 8Y_1^4 \\ Z_3 = 2Y_1Z_1 \end{array}\right\} \quad (8)$$

The total computation cost of doubling using the Jacobian coordinate system $= \mathbf{4M + 6S}$.

V. Proposed Method

Revisiting equation 8, the computation cost is given by

$$3X_1^2 + aZ_1^4 \Rightarrow \mathbf{1M + 2S}$$

But for $a = -3$,

$$3X_1^2 + aZ_1^4 = 3(X_1 + Z_1^2)(X_1 - Z_1^2) \Rightarrow \mathbf{1M + 1S}$$

The computation cost becomes $1M + 1S$. As previously discussed, point doubling is performed during every iteration of the point multiplication operation and hence, for a specific elliptic parameter $\mathbb{E}p(-3, b)$ reduced total computation cost is incurred. For a given point, $P = (X_1 : Y_1 : Z_1)$, $2P = (X_3 : Y_3 : Z_3)$ are given by equation 9.

$$\left.\begin{array}{l} X_3 = A^2 - 2D \\ Y_3 = (D - X_3)A - C^2/2 \\ Z_3 = BZ_1 \end{array}\right\}, \quad (9)$$

where $A = 3(X_1 - Z_1^2)(X_1 + Z_1^2)$, $B = 2Y_1$, $C = B^2, D = CX_1$.

Refer to algorithm 3 for details.

Now, the total computation cost = $\mathbf{4M + 4S}$.

A. Point addition using mixed coordinate system

This paper discusses another method for point addition, which uses both affine and Jacobian coordinate systems. This method is called the mixed coordinate system. In this method, one point is represented in the Jacobian coordinate system, while the other is represented using the affine coordinate system.

The resultant point after addition, is taken in the Jacobian coordinate system. Therefore, for a point $P(X_1, Y_1, Z_1)$ in Jacobian coordinates and $Q(x, y) \equiv Q(X_2, Y_2, Z_2, 1)$ in affine coordinates, where $x_2 = X_2$, $y_2 = Y_2$ and $Z_2 = 1$, $R = P + Q$ is given. [11]

Algorithm 3 Point doubling ($y^2 = x^3 + ax + b$, using Jacobian coordinate for a = 3)

INPUT: $P = (X_1 : Y_1 : Z_1)$ in Jacobian coordinates over $E_p(-3, b) : y^2 = x^3 - 3x + b$

OUTPUT: $2P = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates.

1. **if** $P = \infty$ **then Return**(P)
2. $A \leftarrow 3(X_1 - Z_1^2)(X_1 + Z_1^2)$
3. $B \leftarrow 2Y_1$
4. $Z_3 \leftarrow BZ_1$
5. $C \leftarrow B^2$
6. $D \leftarrow CX_1$
7. $X_3 \leftarrow A^2 - 2D$
8. $Y_3 \leftarrow (D - X_3)A - C^2/2$
9. **Return** $(X_3 : Y_3 : Z_3)$

Algorithm 4 Point addition ($y^2 = x^3 + ax + b$, affine-Jacobian (mixed) coordinates)

INPUT: $P = (X_1 : Y_1 : Z_1)$ in Jacobian coordinate, $Q = (x_2, y_2)$ in affine coordinates on $E(F_p) : y^2 = x^3 + ax + b$.

OUTPUT: $P + Q = (X_3 : Y_3 : Z_3)$ in Jacobian coordinates.

1) **if** $Q = \infty$ **then Return**$(X_1 : Y_1 : Z_1)$.
2) **if** $P = \infty$ **then Return**$(x_2 : y_2 : 1)$.
3) $A \leftarrow Z_1^2$.
4) $B \leftarrow Z_1A$
5) $C \leftarrow X_2A$
6) $D \leftarrow Y_2B$
7) $E \leftarrow C - X_1$
8) $F \leftarrow D - Y_1$
   a) **if** $E == 0$ **then**
      i) **if** $F == 0$ **then** use algorithm 3 to compute $(X_3 : Y_3 : Z_3) = 2(x_2, y_2, 1)$ and **Return**$(X_3 : Y_3 : Z_3)$
9) $G \leftarrow E^2$
10) $H \leftarrow GE$
11) $I \leftarrow X_1G$
12) $X_3 \leftarrow F^2 - (H + 2I)$
13) $Y_3 \leftarrow F(I - X_3) - Y_1H$
14) $Z_3 \leftarrow Z_1E$
15) **Return** $(X_3 : Y_3 : Z_3)$

$$\begin{array}{l} X_3 = (Y_2Z_1^3 - Y_1)^2 - (X_2Z_1 - X_1)^2(X_1 + X_2Z_1^2) \\ Y_3 = (Y_2Z_1^3 - Y_1)(X_1(X_2Z_1^2) - X_3) \\ \qquad\qquad -Y_1(X_2Z_1^2 - X_1)^3 \\ Z_3 = (X_2Z_1^2 - X_1)Z_1 \end{array} \quad (10)$$

The point addition is presented in algorithm 4

DOI: 01.IJRTET.9.1.525

The total computation cost for point addition $= \mathbf{8M} + \mathbf{3S}$.

VI. Results And Comparisons For Nist-Recommended Prime Fields

We have implemented ECC point multiplication in C/C++ using GCC compiler running under a Linux platform, with Intel Pentium processor operating at a clock frequency of $1.73\,GH_z$. The modular multiplication and modular inversion operations are compared for the NIST recommended prime fields because the modular operation is less compute intensive for the specific prime field.

A comparison of different point multiplication techniques, Binary, NAF, w NAF for NIST prime fields is given in figure 1. As can be noticed that the number of addition operations is drastically reduced from Binary to NAF and NAF to w NAF. But higher window sizes require pre computations and more over the number of point doubling operations are irreducible, which is directly proportional to the length of the prime field.

Timing results for the modular multiplication and modular inversion for different prime fields are given in table III, as per the required clock cycles to compute specific operations.

Different coordinate systems are used to eliminate the modular inversion operation, which is a computationally intensive operation. A comparison of various coordinate systems, Affine, Standard Projective, Jacobian and Mixed coordinate system based on the required number of inversion, multiplication and squaring operations along with their execution timings in milli-seconds for scalar point multiplication is given in table IV.

Table III. Timing Of Multiplication And Inversion Operations In Microsecond For NIST Prime Fields

Operation	$\mathbb{F}_{192}$	$\mathbb{F}_{224}$	$\mathbb{F}_{256}$	$\mathbb{F}_{384}$	$\mathbb{F}_{521}$
Multiplication	5.727	9.683	11.678	24.077	35.941
Inversion	88.298	365.573	355.169	1411.323	3012.589

We also propose an algorithm for point doubling for a specific elliptic curve, $\mathbb{E}_p(-3, b)$, which requires less number of multiplication and squaring operations relatively. A point doubling operation, using Jacobian coordinate, requires 0.057 millisecond for $\mathbb{E}_p(a, b)$, but the same requires 0.0458 millisecond for .

Finally, an overall comparison of all of the point multiplication methods for the NIST-recommended prime filed as well as different coordinate systems to perform point addition and point doubling operations is given in table-V.

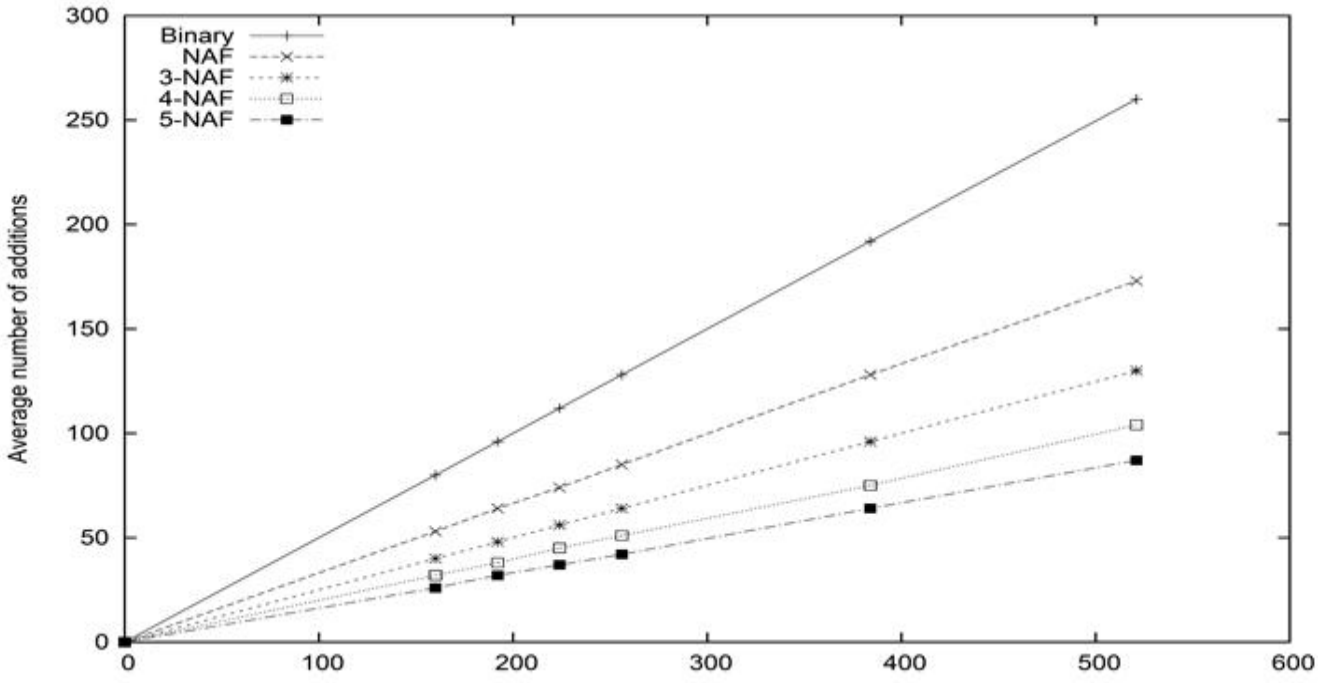

Figure 1. Comparison of various point multiplication techniques

VI. Conclusions

ECC is widely accepted due to its high security per bit compared to traditional RSA public key cryptography. The efficiency of ECC depends on the scalar point multiplication operation. This paper has discussed various point multiplication methods. The computational time for inversion operation is increasing exponentially for higher prime fields compared to multiplication operation. The point multiplication can be efficiently computed using windowed-NAF and by choosing an appropriate window size to meet memory constraints. Different coordinate systems are used to eliminate the expensive inversion operation. The point doubling is efficiently computed using Jacobian coordinate system for $\mathbb{E}_p(-3, b)$ and the point addition is efficiently computed using mixed (affine + Jacobian) coordinate system as per the results shown in table IV.

DOI: 01.IJRTET.9.1.525

TABLE IV. TIMING OF THE DIFFERENT COORDINATE SYSTEM IN **MILLISECOND** FOR SCALAR POINT MULTIPLICA

Coordinate system	Doubling(D)			Addition(A)			Timing									
	I	M	S	I	M	S	$\mathbb{F}_{192}$		$\mathbb{F}_{224}$		$\mathbb{F}_{256}$		$\mathbb{F}_{384}$		$\mathbb{F}_{521}$	
							D	A	D	A	D	A	D	A	D	A
Affine	1	2	2	1	2	1	0.111	0.105	0.404	0.394	0.701	0.69	1.507	1.483	3.156	3.12
Projective $a \in p$	0	7	5	0	12	2	0.068	0.08	0.116	0.135	0.14	0.163	0.288	0.337	0.431	0.503
$a = -3$	0	7	3	0	12	2	0.057	0.08	0.096	0.135	0.116	0.163	0.24	0.337	0.359	0.503
Jacobian $a \in p$	0	4	6	0	12	4	0.057	0.091	0.096	0.154	0.116	0.186	0.24	0.358	0.359	0.575
$a = -3$	0	4	4	0	12	4	0.0458	0.091	0.077	0.154	0.093	0.186	0.192	0.358	0.287	0.575
Mixed	-	-	-	0	8	3		0.062		0.106		0.128		0.264		0.395

TABLE V. TIMING FOR SCALAR POINT MULTIPLICATION FOR NIST PRIME FIELD IN MILLISECONDS

NIST prime field	Point multiplication method	Coordinate system			
		Affine	Std. projective	Jacobian	Mixed
$\mathbb{F}_{192}$	Binary	31.39	18.62	17.52	14.74
	NAF	28.03	16.06	14.61	12.76
	3-NAF	25.82	14.38	12.70	11.45
	4-NAF	25.30	13.98	12.25	11.14
	5-NAF	24.67	13.50	11.70	10.77
$\mathbb{F}_{224}$	Binary	34.6	36.62	34.49	29.12
	NAF	19.6	31.49	28.64	25.09
	3-NAF	12.5	29.06	25.87	23.18
	4-NAF	08.2	27.57	24.17	22.01
	5-NAF	05.0	26.49	22.94	21.17
$\mathbb{F}_{256}$	Binary	267.7	50.56	47.61	40.19
	NAF	238.1	43.55	39.61	34.68
	3-NAF	223.6	40.12	35.71	32
	4-NAF	214.6	38.00	33.29	30.33
	5-NAF	08.4	36.54	31.62	29.18
$\mathbb{F}_{384}$	Binary	363.4	156.8	142.4	124.4
	NAF	768.5	135.2	118.5	107.5
	3-NAF	721.0	124.5	108.0	99.07
	4-NAF	689.9	117.4	100.5	93.52
	5-NAF	673.6	113.7	96.64	90.62
$\mathbb{F}_{521}$	Binary	2460	317.8	304.2	257.4
	NAF	2188	274.0	254.2	223.0
	3-NAF	2054	252.4	229.4	206.0
	4-NAF	1973	239.3	214.5	195.8
	5-NAF	1920	230.8	204.7	189.1

REFERENCES

[1] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[2] M. Amara and A. Siad, "Elliptic curve cryptography and its applications," in *Systems, Signal Processing and their Applications (WOSSPA), 2011 7th International Workshop on*. IEEE, 2011, pp. 247–250.

[3] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Shantz, "Energy analysis of public-key cryptography for wireless sensor networks," in *Pervasive Computing and Communications, 2005. PerCom 2005. Third IEEE International Conference on*. IEEE, 2005, pp. 324–328.

[4] H. Yan and Z. J. Shi, "Studying software implementations of elliptic curve cryptography," in *Information Technology: New Generations, 2006. ITNG 2006. Third International Conference on*, 2006, pp. 78–83.

[5] M. Hassan and M. Benaissa, "Embedded software design of scalable low-area elliptic-curve cryptography," *Embedded Systems Letters, IEEE*, vol. 1, no. 2, pp. 42–45, 2009.

[6] Mohamed N.Hassan and Mohammed Benaissa, "Low area-scalable hardware/software co-design for elliptic curve cryptography," in *New Technologies, Mobility and Security (NTMS), 2009 3rd International Conference on*, 2009, pp. 1–5.

[7] E. Wenger and J. Grossschadl, "An 8-bit avr-based elliptic curve cryptographic risc processor for the internet of things," in *Microarchitecture Workshops (MICROW), 2012 45th Annual IEEE/ACM International Symposium on*, 2012, pp. 39–46.

[8] N. FIPS, "186 digital signature standard," 1994.

[9] D. Karakoyunlu, F. K. Gurkaynak, B. Sunar, and Y. Leblebici, "Efficient and side-channel-aware implementations of elliptic curve cryptosystems over prime fields," *IET information security*, vol. 4, no. 1, pp. 30–43, 2010.

[10] N. Gura, A. Patel, A. Wander, H. Eberle, and S. Shantz, "Comparing elliptic curve cryptography and rsa on 8-bit cpus," *Cryptographic Hardware and Embedded Systems-CHES 2004*, pp. 925–943, 2004.

[11] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to elliptic curve cryptography*. Springer, 2004.

[12] E. Karthikeyan and P. Balasubramaniam, "Improved elliptic curve scalar multiplication algorithm," in *Information and Automation, 2006. ICIA 2006. International Conference on*. IEEE, 2006, pp. 254–257.

[13] A. OZCAN, "Performance analysis of elliptic curve multiplication algorithms for elliptic curve cryptography," Ph.D. dissertation, MIDDLE EAST TECHNICAL UNIVERSITY, 2006.

[14] A. Gutub and S. Arabia, "Remodeling of elliptic curve cryptography scalar multiplication architecture using parallel jacobian coordinate system," *International Journal of Computer Science and Security (IJCSS)*, vol. 4, no. 4, p. 409, 2010.

DOI: 01.IJRTET.9.1.525